Superlinear advantage for exact quantum algorithms

Andris Ambainis[*]
Faculty of Computing, University of Latvia,
Raiņa bulv. 19, Riga, LV-1586, Latvia.
ambainis@lu.lv

ABSTRACT

A quantum algorithm is exact if, on any input data, it outputs the correct answer with certainty (probability 1). A key question is: how big is the advantage of exact quantum algorithms over their classical counterparts: deterministic algorithms. For total Boolean functions in the query model, the biggest known gap was just a factor of 2: PARITY of N inputs bits requires N queries classically but can be computed with $\lceil N/2 \rceil$ queries by an exact quantum algorithm.

We present the first example of a Boolean function $f(x_1, ..., x_N)$ for which exact quantum algorithms have superlinear advantage over the deterministic algorithms. Any deterministic algorithm that computes our function must use N queries but an exact quantum algorithm can compute it with $O(N^{0.8675\ldots})$ queries. A modification of our function gives a similar result for communication complexity: there is a function f which can be computed by an exact quantum protocol that communicates $O(N^{0.8675\ldots})$ quantum bits but requires $\Omega(N)$ bits of communication for classical protocols.

1. INTRODUCTION

Quantum algorithms can be either studied in the *bounded-error* setting (the algorithm must output the correct answer with probability at least 2/3, for every input) or in the *exact* setting (the algorithm must output the correct answer with certainty, for every input). For the bounded-error case, there are many quantum algorithms that are better than classical algorithms ([30, 16, 2, 15] and many others).

It is much more difficult to come up with exact quantum algorithms that outperform classical algorithms. The requirement that the algorithm's answer must always be correct is very constraining: it means that, in the algorithm's final state, we cannot have even very small non-zero amplitudes for the basis states that correspond to an incorrect answer. Arranging the algorithm's transformations so that this requirement is satisfied for all of the possible inputs has been a very challenging problem.

We consider computing Boolean functions in the query model. Let $Q_E(f)$ ($Q_2(f)$) be the smallest number of queries in an exact (bounded-error) quantum algorithm that computes f and $D(f)$ be the smallest number of queries in a deterministic algorithm that computes f. For total Boolean functions, the biggest gap between $Q_E(f)$ and $D(f)$ has been achieved for the PARITY of N input bits. A modification of Deutsch's algorithm [13] discovered by Cleve et al. [11] can compute PARITY of 2 input bits exactly with just 1 quantum query. This immediately implies that PARITY of N bits can be computed with $\lceil N/2 \rceil$ queries. In contrast, deterministic algorithms need N queries to compute PARITY.

Bigger speedups are known for partial functions. For example, Simon's algorithm [31] can be made exact [6]. This gives a partial function $f(x_1, \ldots, x_N)$ for which $Q_E(f) = O(\log N)$ and $D(f) = \Omega(\sqrt{N})$. The value of this function $f(x_1, \ldots, x_N)$ is, however, defined only for very small fraction of all inputs $(x_1, \ldots, x_N)$.

Many attempts have been made to come up with exact quantum algorithms for total functions but the best results have been algorithms that achieve the same separation as for the PARITY function: $Q_E(f) = N/2$ vs. $D(f) = N$ (either by using the parity algorithm by a subroutine (e.g. [32]) or by different methods [22]).

In this paper, we give the first separation between $Q_E(f)$ and $D(f)$ that is more than a factor of 2. Namely, we obtain $Q_E(f) = O(D(f)^{0.8675\ldots})$ for a sequence of functions f (with $D(f) \to \infty$).

The sequence of functions is as follows. We start with the function $NE(x_1, x_2, x_3)$ defined by

- $NE(x_1, x_2, x_3) = 1$ if $x_i \neq x_j$ for some $i, j \in \{1, 2, 3\}$;
- $NE(x_1, x_2, x_3) = 0$ if $x_1 = x_2 = x_3$.

We define $NE^0(x_1) = x_1$ and

$$NE^d(x_1, \ldots, x_{3^d}) = NE(NE^{d-1}(x_1, \ldots, x_{3^{d-1}}),$$

[*]Supported by ESF project 1DP/1.1.1.2.0/09/APIA/VIAA/044 and the European Commission under the project QCS (Grant No. 255961)

$$NE^{d-1}(x_{3^{d-1}+1}, \ldots, x_{2\cdot 3^{d-1}}), NE^{d-1}(x_{2\cdot 3^{d-1}+1}, \ldots, x_{3^d}))$$

for $d > 0$. This sequence of functions has been known as a candidate for a superlinear separation between $D(f)$ and $Q_E(f)$ for a long time (since at least [10]). The reason for that is the relationship between $Q_E(f)$ and the polynomial degree of f [4].

Let $deg(f)$ be the degree of the unique multilinear polynomial that is equal to $f(x_1, \ldots, x_N)$. Then [4], $D(f) \geq Q_E(f) \geq deg(f)/2$. If we also have $D(f) = deg(f)$ (which is true for many functions f), this implies that $Q_E(f) \geq D(f)/2$.

To obtain a bigger gap between $D(f)$ and $Q_E(f)$, we should start with f which has $D(f) > deg(f)$. NE^d has this property. We have $D(NE) = 3$ and $deg(NE) = 2$ which implies $D(NE^d) = 3^d$ and $deg(NE^d) = 2^d$ [24]. (There are other constructions of functions with $D(f) > deg(f)$, by taking a different basis function f instead of NE and iterating it in the same way [20, 1].)

Buhrman and de Wolf [10] observed that this means the following. If we determine $Q_E(f)$, we will either get $Q_E(f) = o(D(f))$ or $deg(f) = o(Q_E(f))$ (showing that the degree lower bound on $Q_E(f)$ is not tight). The second of those results was obtained in [1] which showed that $Q_E(NE^d) \geq Q_2(NE^d) = \Omega(2.121...^d)$. For bounded error algorithms, the work on negative adversary bound [18] and span programs [29, 27, 28] resulted in the conclusion that this bound is optimal: $Q_2(NE^d) = \Theta(2.121...^d)$.

Even though the function NE^d is quite well known, no positive results on its exact complexity have been obtained. In this paper, we provide the first nontrivial exact quantum algorithm for NE^d, showing that $Q_E(NE^d) = O(2.593...^d) = O(D(NE^d)^{0.8675...})$.

Main ideas. The main ideas behind our algorithm are as follows. We create an algorithm in which one basis state has amplitude 1 if $NE^d(x_1, \ldots, x_{3^d}) = 0$ and an amplitude $\alpha < 1$ if $NE^d(x_1, \ldots, x_{3^d}) = 1$. The algorithm is constructed so that α is the same for all $(x_1, \ldots, x_{3^d})$ with $NE^d(x_1, \ldots, x_{3^d}) = 1$. The construction is by induction: we use the algorithm of this type for NE^{d-1} as a subroutine to construct the algorithm for NE^d.

Each such induction step decreases the difference between the $NE^d = 0$ and $NE^d = 1$ cases, bringing α closer to 1. To compensate for that, we interleave the induction steps with a form of quantum amplitude amplification [7] which increases the difference between α and 1. At the end, we perform the amplitude amplification again, to construct an algorithm that perfectly distinguishes between the two cases.

Communication complexity. Besides query complexity, our result also applies to the setting of communication complexity (in the standard two-party communication model). Then, we get a total function $f(y_1, \ldots, z_N)$ which can be computed by an exact quantum protocol that communicates $O(N^{0.865...} \log N)$ quantum bits but requires $\Omega(N)$ bits for classical protocols in a variety of models (deterministic, bounded-error probabilistic and nondeterministic protocol) [33]. Previously, it was known that a classical protocol that communicates k bits can be converted into a quantum protocol that uses shared entanglement and $k/2$ quantum bits of communication (via quantum teleportation [5]). However, no provable gap of any size between exact quantum and deterministic communication complexity was known for a total function in the case when the quantum protocol does not have shared entanglement. (A communication complexity version of the Deutsch-Jozsa problem [14, 8] gives an exponential gap for a partial function.)

2. DEFINITIONS

We assume familiarity with the standard notions of quantum states and transformations [23]. We now briefly define the quantum query model, to synchronize the notation with the reader.

We assume that the task is to compute a Boolean function $f(x_1, \ldots, x_N)$, $x_1, \ldots, x_N \in \{0, 1\}$. We consider a Hilbert space $\mathcal{H}$ with basis states $|i, j\rangle$ for $i \in \{0, 1, \ldots, N\}$ and $j \in \{1, \ldots, M\}$ (where M can be chosen arbitrarily). We define that a query Q is the following transformation:

- $Q|0, j\rangle = |0, j\rangle$;
- $Q|i, j\rangle = (-1)^{x_i}|i, j\rangle$ for $i \in \{1, 2, \ldots, N\}$.

A quantum query algorithm $\mathcal{A}$ consists of a sequence of transformations U_0, Q, U_1, ..., U_{k-1}, Q, U_k where Q are *queries* and U_i are arbitrary unitary transformations that do not depend on $x_1, \ldots, x_N$. The algorithm starts with a fixed starting state $|\psi_{start}\rangle$ and performs the transformations. This leads to the final state

$$|\psi_{end}\rangle = U_k Q U_{k-1} \ldots U_1 Q U_0 |\psi_{start}\rangle.$$

We then measure this state and interpret the result as a binary value $y \in \{0, 1\}$. (That is, we define some of possible results as corresponding to $y = 0$ and others as corresponding to $y = 1$.)

An algorithm $\mathcal{A}$ computes $f(x_1, \ldots, x_N)$ *exactly* if, for every $x_1, \ldots, x_N \in \{0, 1\}$, the obtained value y is always equal to $f(x_1, \ldots, x_N)$.

3. RESULTS AND PROOFS

3.1 Results

Our main result is

THEOREM 1.

$$Q_E(NE^d) = O(2.593...^d).$$

Since $D(NE^d) = 3^d$, this means that

$$Q_E(NE^d) = O(D(NE^d)^{0.8675...}),$$

giving a polynomial separation between $D(f)$ and $Q_E(f)$. One can also show that $R_2(NE^d)$, the probabilistic query complexity of NE^d, is equal 3^d, as well. Therefore, we also get the same separation between $D(f)$ and $R_2(f)$.

3.2 Framework

In this subsection, we develop the tools for proving Theorem 1. One of our main ideas is to construct algorithms which produce a final state in which one amplitude takes one of two values - depending on $f(x_1, \ldots, x_N)$:

Definition 1. *Let $p \in [-1, 1]$. A quantum query algorithm $\mathcal{A}$ p-computes a function $f(x_1, \ldots, x_N)$ if:*

(a) $\mathcal{A}|\psi_{start}\rangle = |\psi_{start}\rangle$ whenever $f(x_1, \ldots, x_N) = 0$;

(b) if $f(x_1, \ldots, x_N) = 1$, then

$$\mathcal{A}|\psi_{start}\rangle = p|\psi_{start}\rangle + \sqrt{1-p^2}|\psi\rangle$$

for some $|\psi\rangle : |\psi\rangle \perp |\psi_{start}\rangle$ (where $|\psi\rangle$ may depend on $x_1, \ldots, x_n$);

We note that p-computing a function becomes easier when p increases. The easiest case is $p = 1$ when any function can be 1-computed in a trivial way by performing the identity transformation I on $|\psi_{start}\rangle$. For $p = 0$, an algorithm $\mathcal{A}$ that 0-computes f is also an exact algorithm for f in the usual sense because we can measure whether the final state of $\mathcal{A}$ is $|\psi_{start}\rangle$ is $|\psi_{start}\rangle$ or orthogonal to $|\psi_{start}\rangle$ and output 0 in the first case and 1 in the second case.

We also have

Lemma 1. *If an algorithm $\mathcal{A}$ p-computes f with k queries, there is an algorithm $\mathcal{A}'$ that p'-computes f with k queries, for any $p' > p$.*

Proof: We enlarge the state space of the algorithm by adding a new basis state $|0, j\rangle$ which is left unchanged by queries Q. We extend all U_i to the enlarged state space by defining $U_i|0, j\rangle = |0, j\rangle$ and change the starting state to $|\psi'_{start}\rangle = \cos\alpha|\psi_{start}\rangle + \sin\alpha|0, j\rangle$.

If $f = 0$, we have $\mathcal{A}|\psi_{start}\rangle = |\psi_{start}\rangle$ and, hence, $\mathcal{A}|\psi'_{start}\rangle = |\psi'_{start}\rangle$.

If $f = 1$, then $\mathcal{A}|\psi'_{start}\rangle = p'|\psi'_{start}\rangle + \sqrt{1-(p')^2}|\psi'\rangle$ ($|\psi'\rangle \perp |\psi'_{start}\rangle$) for $p' = \langle\psi'_{start}|\mathcal{A}|\psi'_{start}\rangle$. We have

$$p' = p\cos^2\alpha + \sin^2\alpha.$$

By varying α over the interval $[0, \frac{\pi}{2}]$, we can achieve any value of p' between $p' = p$ (for $\alpha = 0$) and $p' = 1$ (for $\alpha = \frac{\pi}{2}$). $\square$

If we have an algorithm $\mathcal{A}$ that p-computes NE^{i-1}, we can use it to build an algorithm $\mathcal{A}'$ that p'-computes NE^i, through the following lemma.

Lemma 2. *If an algorithm $\mathcal{A}$ p-computes NE^{d-1} with k queries, there is an algorithm $\mathcal{A}'$ that p'-computes NE^d with $2k$ queries, for $p' = 1 - \frac{4(1-p)^2}{9}$.*

Proof: We assume that the algorithm $\mathcal{A}$ consists of transformations U_0, Q, U_1, $\ldots$, U_{k-1}, Q, U_k, in a Hilbert space $\mathcal{H}$ with basis states $|i, j\rangle$, $i \in \{0, 1, \ldots, 3^{d-1}\}$, $j \in \{1, \ldots, M\}$. Let $|\psi_{start}\rangle$ be the starting state of $\mathcal{A}$.

We consider a Hilbert space $\mathcal{H}'$ with basis states $|i, j\rangle$, $i \in \{0, 1, \ldots, 3^d\}$, $j \in \{1, \ldots, 3M\}$. In this Hilbert space, we can compute $NE^{d-1}(x_1, \ldots, x_{3^{d-1}})$, $NE^{d-1}(x_{3^{d-1}+1}, \ldots, x_{2\cdot 3^{d-1}})$, $NE^{d-1}(x_{2\cdot 3^{d-1}+1}, \ldots, x_{3^d})$ in parallel, in the following way.

Let $\mathcal{H}_l$ ($l \in \{1, 2, 3\}$) be the subspace spanned by the basis states $|0, j\rangle$, $j \in \{(l-1)M + 1, \ldots lM\}$ and $|i, j\rangle$, $i \in \{(l-1)3^{d-1} + 1, \ldots l3^{d-1}\}$, $j \in \{1, \ldots, M\}$. We have an isomorphism $V_l : \mathcal{H}_l \to \mathcal{H}$ defined by

$$V_l|0, (l-1)M + j\rangle = |0, j\rangle,$$

$$V_l|(l-1)3^{d-1} + i, j\rangle = |i, j\rangle.$$

We define $U_i^{(l)} = (V_l)^\dagger U_i V_l$ and $|\psi_{start}^{(l)}\rangle = (V_l)^\dagger|\psi_{start}\rangle$. Then, the sequence of transformations

$$U_0^{(l)}, Q, U_1^{(l)}, \ldots, U_{k-1}^{(l)}, Q, U_k^{(l)}$$

with the starting state $|\psi_{start}^{(l)}\rangle$ is a quantum algorithm p-computing the function $NE^{d-1}(x_{(l-1)3^{d-1}+1}, \ldots, x_{l\cdot 3^{d-1}})$.

Let U'_i be a transformation which is equal to $U_i^{(l)}$ on $\mathcal{H}_l$, for each $l \in \{1, 2, 3\}$. Then, the sequence of transformations U'_0, Q, U'_1, $\ldots$, U'_{k-1}, Q, U'_k can be used to p-compute any of $NE^{d-1}((l-1)x_{3^{d-1}+1}, \ldots, x_{l\cdot 3^{d-1}})$, for $l \in \{1, 2, 3\}$, depending on the starting state. Let

$$V = U'_k Q U'_{k-1} \ldots, U'_1 Q U'_0$$

denote the product of these transformations.

Let T be the unitary transformation defined by:

- $T|\psi'_{start}\rangle = |\psi'_{start}\rangle$ for $|\psi'_{start}\rangle = \sum_{l=1}^{3} \frac{1}{\sqrt{3}}|\psi_{start}^{(l)}\rangle$;
- $T|\psi\rangle = -|\psi\rangle$ for any $|\psi\rangle$ that is a linear combination of $|\psi_{start}^{(1)}\rangle$, $|\psi_{start}^{(2)}\rangle$, $|\psi_{start}^{(3)}\rangle$ and satisfies $|\psi\rangle \perp |\psi'_{start}\rangle$;
- $T|\psi\rangle = |\psi\rangle$ for any $|\psi\rangle$ that is perpendicular to all of $|\psi_{start}^{(1)}\rangle$, $|\psi_{start}^{(2)}\rangle$, $|\psi_{start}^{(3)}\rangle$.

We take the algorithm $\mathcal{A}'$ which performs the transformation $\mathcal{A}' = V^{-1}TV$, on the starting state $|\psi'_{start}\rangle$. We claim that this algorithm p'-computes the function NE^d.

To prove it, we consider two cases.

Case 1: $NE^d(x_1, \ldots, x_{3^d}) = 0$.

This means that we have one of two subcases.

Case 1a: $NE^{d-1}(x_{(l-1)3^{d-1}+1}, \ldots, x_{l\cdot 3^{d-1}}) = 0$ for all $l \in \{1, 2, 3\}$.

Then, given a starting state $|\psi_{start}^{(l)}\rangle$, V performs the algorithm for p-computing $NE^{d-1}(x_{(l-1)3^{d-1}+1}, \ldots, x_{l\cdot 3^{d-1}}) = 0$. This means that $V|\psi_{start}^{(l)}\rangle = |\psi_{start}^{(l)}\rangle$ for all $l \in \{1, 2, 3\}$

and, hence, $V|\psi'_{start}\rangle = |\psi'_{start}\rangle$. Since $T|\psi'_{start}\rangle = |\psi'_{start}\rangle$ (by the definition of T), we get that $\mathcal{A}|\psi'_{start}\rangle = |\psi'_{start}\rangle$.

Case 1b: $NE^{d-1}((l-1)x_{3^{d-1}+1}, \ldots, x_{l\cdot 3^{d-1}}) = 1$ for all $l \in \{1,2,3\}$.

Then, $V|\psi^{(l)}_{start}\rangle = p|\psi^{(l)}_{start}\rangle + \sqrt{1-p^2}|\psi^{(l)}\rangle$ where $|\psi^{(l)}\rangle \in \mathcal{H}_l$ and $|\psi^{(l)}\rangle \perp |\psi^{(l)}_{start}\rangle$. Trivially, we also have $|\psi^{(l)}\rangle \perp |\psi^{(l')}_{start}\rangle$ for $l \neq l'$ (since $|\psi^{(l)}\rangle \in \mathcal{H}_l$, $|\psi^{(l')}_{start}\rangle \in \mathcal{H}_{l'}$ and $\mathcal{H}_l \perp \mathcal{H}_{l'}$). This means that

$$V|\psi'_{start}\rangle = \frac{1}{\sqrt{3}}\sum_{l=1}^{3} p|\psi^{(l)}_{start}\rangle + \frac{1}{\sqrt{3}}\sqrt{1-p^2}\sum_{l=1}^{3}|\psi^{(l)}\rangle.$$

Applying T to this state does not change it because the first component is equal to $p|\psi'_{start}\rangle$ and the second component is orthogonal to all $|\psi^{(l)}_{start}\rangle$. Hence,

$$V^{-1}TV|\psi'_{start}\rangle = V^{-1}V|\psi'_{start}\rangle = |\psi'_{start}\rangle.$$

Case 2: $NE^d(x_1, \ldots, x_{3^d}) = 1$.

In this case, we have to prove that $\langle\psi'_{start}|V^{-1}TV|\psi'_{start}\rangle = p'$.

We express $V|\psi'_{start}\rangle = \alpha|\psi_+\rangle + \beta|\psi_-\rangle$ where $T|\psi_+\rangle = |\psi_+\rangle$ and $T|\psi_-\rangle = -|\psi_-\rangle$ and α, β satisfy $|\alpha|^2 + |\beta|^2 = 1$. Then,

$$\langle\psi'_{start}|V^{-1}TV|\psi'_{start}\rangle = |\alpha|^2 - |\beta|^2 = 1 - 2|\beta|^2.$$

To calculate β, we consider two subcases:

Case 2a: $NE^{d-1}(x_{(l-1)3^{d-1}+1}, \ldots, x_{l\cdot 3^{d-1}}) = 1$ for exactly one of $l \in \{1,2,3\}$.

Without loss of generality, we assume that this is $l = 1$. Then,

$$V|\psi'_{start}\rangle = \frac{1}{\sqrt{3}}\left(p|\psi^{(1)}_{start}\rangle + \sqrt{1-p^2}|\psi^{(l)}\rangle + |\psi^{(2)}_{start}\rangle + |\psi^{(3)}_{start}\rangle\right).$$

We have

$$\beta|\psi_-\rangle = \frac{2p-2}{3\sqrt{3}}|\psi^{(1)}_{start}\rangle + \frac{1-p}{3\sqrt{3}}|\psi^{(2)}_{start}\rangle + \frac{1-p}{3\sqrt{3}}|\psi^{(3)}_{start}\rangle,$$

$$|\beta|^2 = \frac{(2p-2)^2}{27} + 2\frac{(1-p)^2}{27} = \frac{2(1-p)^2}{9}.$$

Case 2b: $NE^{d-1}(x_{(l-1)3^{d-1}+1}, \ldots, x_{l\cdot 3^{d-1}}) = 1$ for exactly two of $l \in \{1,2,3\}$.

Without loss of generality, we assume that these are $l = 1$ and $l = 2$. Then,

$$V|\psi'_{start}\rangle = \frac{1}{\sqrt{3}}\left(p|\psi^{(1)}_{start}\rangle + \sqrt{1-p^2}|\psi^{(l)}\rangle\right.$$

$$\left.+p|\psi^{(2)}_{start}\rangle + \sqrt{1-p^2}|\psi^{(2)}\rangle + |\psi^{(2)}_{start}\rangle\right).$$

We have

$$\beta|\psi_-\rangle = \frac{p-1}{3\sqrt{3}}|\psi^{(1)}_{start}\rangle + \frac{p-1}{3\sqrt{3}}|\psi^{(2)}_{start}\rangle + \frac{2-2p}{3\sqrt{3}}|\psi^{(3)}_{start}\rangle$$

and, similarly to the previous case, we get $|\beta|^2 = \frac{2(1-p)^2}{9}$. □

Applying Lemma 2 results in an algorithm with $p' > p$. Applying it several times degrades p even further (that is, p becomes closer and closer to 1). To compensate for that, we have the following lemma for improving p (by adapting quantum amplitude amplification [7] to our setting).

Lemma 3. *If an algorithm $\mathcal{A}$ p-computes a function f with k queries, for $p = \cos\alpha$, there is an algorithm $\mathcal{A}'$ that p'-computes f with ck queries, for $p' = \cos c\alpha$.*

Proof: Let $|\psi_{start}\rangle$ be the starting state of $\mathcal{A}$. Let T be the transformation defined by $T|\psi_{start}\rangle = |\psi_{start}\rangle$ and $T|\psi\rangle = -|\psi\rangle$ for all $|\psi\rangle : |\psi\rangle \perp |\psi_{start}\rangle$.

The new algorithm $\mathcal{A}'$ has the same starting state $|\psi_{start}\rangle$ and consists of transformations $V_1, T, V_2, T, \ldots, T, V_c$ where $V_i = \mathcal{A}$ for odd i and $V_i = \mathcal{A}^{-1}$ for even i. Since $\mathcal{A}$ and $\mathcal{A}^{-1}$ both use k queries and T can be performed with no queries, the new algorithm $\mathcal{A}'$ uses ck queries.

If $f(x_1, \ldots, x_N) = 0$, then (by definition 1), $\mathcal{A}|\psi_{start}\rangle = |\psi_{start}\rangle$. Since this also means $\mathcal{A}^{-1}|\psi_{start}\rangle = |\psi_{start}\rangle$, we get that $\mathcal{A}'|\psi_{start}\rangle = |\psi_{start}\rangle$.

In the $f(x_1, \ldots, x_N) = 1$ case, we have $\mathcal{A}|\psi_{start}\rangle = p|\psi_{start}\rangle + \sqrt{1-p^2}|\psi_1\rangle$ for some $|\psi_1\rangle \perp |\psi_{start}\rangle$. We can also express

$$\mathcal{A}^{-1}T\mathcal{A}|\psi_{start}\rangle = \cos\beta|\psi_{start}\rangle + \sin\beta|\psi_2\rangle$$

for some $|\psi_2\rangle : |\psi_2\rangle \perp |\psi_{start}\rangle$ and $\beta \in [0,\pi]$. (We will show that $\beta = 2\alpha$.) We define

$$|\varphi_{start}\rangle = \cos\alpha|\psi_{start}\rangle + \sin\alpha|\psi_1\rangle,$$

$$|\varphi_2\rangle = \sin\alpha|\psi_{start}\rangle - \cos\alpha|\psi_1\rangle.$$

Then, $|\varphi_{start}\rangle \perp |\varphi_2\rangle$.

Claim 1. *$\mathcal{A}$ maps $|\psi_{start}\rangle$ and $|\psi_2\rangle$ as follows:*

$$\mathcal{A}|\psi_{start}\rangle = |\varphi_{start}\rangle,$$

$$\mathcal{A}|\psi_2\rangle = |\varphi_2\rangle.$$

Proof: The first equality is immediate. For the second equality, we consider the three vectors

$$\mathcal{A}|\psi_{start}\rangle = |\varphi_{start}\rangle,$$

$$T\mathcal{A}|\psi_{start}\rangle = \cos\alpha|\psi_{start}\rangle - \sin\alpha|\psi_1\rangle,$$

$$|\varphi_2\rangle = \sin\alpha|\psi_{start}\rangle - \cos\alpha|\psi_1\rangle.$$

These three vectors are in the same plane. Hence, $\mathcal{A}^{-1}$ maps them to three vectors in the same plane and preserves the angles between the three vectors (since $\mathcal{A}^{-1}$ is unitary). The first two vectors are mapped to $|\psi_{start}\rangle$ and $\cos\beta|\psi_{start}\rangle + \sin\beta|\psi_2\rangle$, respectively. Hence, $\mathcal{A}^{-1}$ must map the third vector ($|\varphi_2\rangle$) to a vector in the same plane that has the same angles with these two vectors - that is, to $|\psi_2\rangle$. □

We now define $\mathcal{A}_j$ as the algorithm that consists of the first steps of $\mathcal{A}'$, from V_1 to V_j. Then, $\mathcal{A}' = \mathcal{A}_c$.

Lemma 3 follows by substituting $j = c$ into

CLAIM 2.

$$\mathcal{A}_j|\psi_{start}\rangle = \begin{cases} \cos j\alpha|\psi_{start}\rangle + \sin j\alpha|\psi_1\rangle & \text{if } j \text{ is odd} \\ \cos j\alpha|\psi_{start}\rangle - \sin j\alpha|\psi_2\rangle & \text{if } j \text{ is even} \end{cases}$$

Proof: By induction. The base case, $j = 1$, immediately follows from the definition of $|\psi_1\rangle$.

The inductive case is slightly different for even j and for odd j. If j is even, $\mathcal{A}_j = \mathcal{A}^{-1}T\mathcal{A}_{j-1}$. From the inductive assumption and the definition of T,

$$T\mathcal{A}_{j-1}|\psi_{start}\rangle = \cos(j-1)\alpha|\psi_{start}\rangle - \sin(j-1)\alpha|\psi_1\rangle.$$

We can express this state as

$$\cos j\alpha|\varphi_{start}\rangle - \sin j\alpha|\varphi_2\rangle.$$

Therefore,

$$\mathcal{A}^{-1}T\mathcal{A}_{j-1}|\psi_{start}\rangle = \cos j\alpha|\psi_{start}\rangle - \sin j\alpha|\psi_2\rangle.$$

If j is odd, $\mathcal{A}_j = \mathcal{A}T\mathcal{A}_{j-1}$. Then,

$$T\mathcal{A}_{j-1}|\psi_{start}\rangle = \cos(j-1)\alpha|\psi_{start}\rangle + \sin(j-1)\alpha|\psi_2\rangle,$$

$$\mathcal{A}T\mathcal{A}_{j-1}|\psi_{start}\rangle = \cos(j-1)\alpha|\varphi_{start}\rangle + \sin(j-1)\alpha|\varphi_2\rangle$$

$$= \cos j\alpha|\psi_{start}\rangle + \sin j\alpha|\psi_1\rangle.$$

□ □

As a special case of Lemma 3, we obtain

COROLLARY 1. *If an algorithm $\mathcal{A}$ p-computes NE^d with k queries, there is an algorithm $\mathcal{A}'$ that p'-computes NE^d with $2k$ queries, for $p' = 2p^2 - 1$.*

Proof: We set $c = 2$ in Lemma 3. Then $p' = \cos 2(\arccos p) = 2p^2 - 1$. □

3.3 Algorithm for NE^d

It is easy to see that

LEMMA 4. *If there is an algorithm $\mathcal{A}$ that -1-computes NE^t with k queries, there is an algorithm $\mathcal{A}_l$ that -1-computes NE^{tl} with k^l queries for all $l \geq 1$.*

Proof: By induction. The algorithm $\mathcal{A}$ itself forms the base case, for $l = 1$.

For the inductive case, we can obtain the function NE^{tl} by taking the function $NE^{t(l-1)}$ and, instead of each variable, substituting a function NE^t on a block of 3^t new variables. Therefore, we can take the algorithm $\mathcal{A}_{l-1}$ that computes $NE^{t(l-1)}$ with k^{l-1} queries and, instead of each query, substitute the algorithm $\mathcal{A}$ that performs $\mathcal{A}|\psi\rangle = |\psi\rangle$ if $NE^t = 0$ for the corresponding group of 3^t variables and $\mathcal{A}|\psi\rangle = -|\psi\rangle$ if $NE^t = 1$.

In the resulting algorithm $\mathcal{A}_l$, each of k^{l-1} queries of $\mathcal{A}_l$ is replaced by a sequence of transformations that involves k queries. Therefore, the total number of queries for $\mathcal{A}_l$ is $k^{l-1}k = k^l$. □

Moreover, we get that $Q_E(NE^{tl}) \leq k^l$, since an algorithm that -1-computes NE^{tl} can be transformed into an algorithm that 0-computes NE^{tl}, with no increase in the number of queries (Lemma 1) and an algorithm for 0-computing f can be used to compute f exactly. Therefore, we have

COROLLARY 2. *If there is an algorithm $\mathcal{A}$ that -1-computes NE^t with k queries, then $Q_E(NE^d) = O(k^{d/t})$.*

It remains to find a base algorithm that -1-computes NE^d with less than 3^d queries. We give two such constructions.

Construction 1:

1. $NE^0(x_1) = x_1$ can be -1-computed with 1 query by just performing a query $|1\rangle \to (-1)^{x_1}|1\rangle$.
2. Applying Lemma 2 with $p = -1$ gives that NE^1 can be $-\frac{7}{9}$ computed with 2 queries.
3. Applying Lemma 2 with $p = -\frac{7}{9}$ gives that NE^2 can be p'-computed with 4 queries for $p' = -\frac{295}{729}$.
4. Applying Lemma 1 gives that NE^2 can be 0-computed with 4 queries.
5. Applying Corollary 1 with $p = 0$ gives that NE^2 can be -1-computed with 8 queries.

By Corollary 2, this means that $Q_E(NE^d) = O(8^{d/2}) = O(2.828...^d)$. The exponent can be improved by using a more complicated base construction with a larger number of steps.

Construction 2:

1. Start with an algorithm that $-\frac{295}{729}$-computes NE^2 with 4 queries (from Construction 1).
2. Applying Lemma 2 with $p = -\frac{295}{729}$ gives that NE^3 can be p'-computed with 8 queries for $p' = \frac{588665}{4782969} = 0.123075....$
3. Applying Corollary 1 gives that NE^3 can be p'-computed with 16 queries for $p' = -0.969704....$
4. Applying Lemma 2 3 times gives that NE^6 can be p'-computed with 128 queries for $p' = 0.223874....$
5. Applying Corollary 1 gives that NE^6 can be p'-computed with 256 queries for $p' = -0.8997602....$
6. Applying Lemma 2 2 times gives that NE^8 can be p'-computed with 1024 queries for $p' = -0.15353....$
7. Applying Lemma 1 gives that NE^8 can be also 0-computed with 1024 queries and applying Lemma 1 gives that NE^8 can be -1-computed with 2048 queries,

This means that $Q_E(NE^d) = O(2048^{d/8}) = O(2.593...^d)$.

3.4 Implications for communication complexity

We consider the problem of computing a function

$$f(y_1, \ldots, y_N, z_1, \ldots, z_N)$$

in the communication complexity setting in which one party (Alice) holds $y_1, \ldots, y_N$ and another party (Bob) holds z_1, …, z_N. The task is to compute $f(y_1, \ldots, z_N)$ with the minimum communication. (For a review on quantum communication complexity, see [9].)

An exact quantum algorithm is an algorithm that communicates k qubits and, after communicating those k qubits, both parties output answers that are always equal to $f(y_1, \ldots, z_N)$. The communication complexity counterpart of Theorem 1 is the following theorem (due to Ronald de Wolf [33]):

THEOREM 2. *[33] There exists $f(y_1, \ldots, y_N, z_1, \ldots, z_N)$ such that*

1. *f can be computed exactly by a quantum protocol that communicates $O(N^{0.8675\ldots} \log N)$ quantum bits;*
2. *Any deterministic protocol (or bounded-error probabilistic protocol or nondeterministic protocol) computing f communicates at least $\Omega(N)$ bits.*

Proof: Let $N = 3^d$. We define

$$f(y_1, \ldots, z_N) = NE^d(y_1 \wedge z_1, \ldots, y_N \wedge z_N).$$

As shown in [8], existence of a T query quantum algorithm for $g(x_1, \ldots, x_N)$ implies the existence of a quantum protocol for $g(y_1 \wedge z_1, \ldots, y_N \wedge z_N)$ that communicates $O(T \log N)$ quantum bits. (Alice runs the query algorithm and implements each query through $O(\log N)$ quantum bits of communication with Bob.) This implies the first part of the theorem.

The second part follows by a reduction from the set disjointness problem [19, 26]: $DISJ(y_1, \ldots, z_N) = 1$ if the sets $\{i : y_i = 1\}$ and $\{i : z_i = 1\}$ are disjoint and $DISJ(y_1, \ldots, z_N) = 0$ if these two sets are not disjoint. This is equivalent to

$$DISJ(y_1, \ldots, z_N) = NOT\ OR(y_1 \wedge z_1, \ldots, y_N \wedge z_N).$$

THEOREM 3. *[19, 26] Any deterministic protocol (or bounded-error probabilistic protocol or nondeterministic protocol) for computing $DISJ$ requires communicating $\Omega(N)$ bits, even if it is promised that $y_1, \ldots, y_N$ and $z_1, \ldots, z_N$ are such that there is at most one $i : y_i = z_i = 1$.*

We have $NE^d(x_1, \ldots, x_N) = NOT\ OR(x_1, \ldots, x_N)$ for inputs $(x_1, \ldots, x_N)$ with at most one $i : x_i = 1$. Hence, Theorem 3 implies the second part of Theorem 2. □

4. CONCLUSION AND OPEN PROBLEMS

We have shown that, for the iterated 3-bit non-equality function, $Q_E(NE^m) = O(D(NE^m)^{0.8675\ldots})$. This is the first example of a gap between $Q_E(f)$ and $D(f)$ that is more than a factor of 2. We think that there are more exact quantum algorithms that are waiting to be discovered.

Some possible directions for future work are:

1. Can we improve on $Q_E(NE^d) = O(2.593\ldots^d)$, either by using our methods or in some other way?
2. If $Q_E(NE^d)$ is asymptotically larger than $Q_2(NE^d) = \Theta(2.121\ldots^d)$, can we prove that? There are cases in which we can show lower bounds on $Q_E(f)$ that are asymptotically larger than $Q_2(f)$ [4] but the typical lower bound methods are based on polynomial degree $deg(f)$ and, thus, are unlikely to apply to NE^d for which $deg(NE^d) = 2^d$ is smaller than both $Q_E(NE^d)$ and $Q_2(NE^d)$.
3. How big can the gap between $Q_E(f)$ and $D(f)$ be? Currently, the best lower bound is $Q_E(f) = \Omega(D(f)^{1/3})$ [21].
4. There are other examples of functions $f^d(x_1, \ldots, x_{n^d})$ with $deg(f) = O(D(f)^c)$, $c < 1$ obtained by iterating different basis functions $f(x_1, \ldots, x_n)$ [20, 1] Can we show that $Q_E(f^d) = O(D(f^d)^c)$, $c < 1$ for those functions as well?
5. Computer experiments [22] show that exact quantum algorithms can be quite common: $Q_E(f) < D(f)$ for many functions f on a small number of variables (and, in most cases, an algorithm that uses $Q_E(f)$ queries cannot be produced simply by using the PARITY algorithm).

   Can we develop a general framework that will be able to produce exact algorithms for a variety of functions $f(x_1, \ldots, x_N)$?
6. What is $Q_E(f)$ for a random n-variable Boolean function $f(x_1, \ldots, x_N)$? We know that $Q_E(f) \leq N$ (trivially) and $Q_E(f) \geq Q_2(f) = \frac{N}{2} + o(N)$ [3]. Which of these two bounds is optimal?

   A similar question was recently resolved for $Q_2(f)$, by showing that $Q_2(f) = \frac{N}{2} + o(N)$ for a random f, with a high probability [3].

Acknowledgment. I thank Aleksandrs Belovs for useful comments about an earlier draft of this paper and Ronald de Wolf for pointing out Theorem 2 and its proof.

5. REFERENCES

[1] A. Ambainis. Polynomial degree vs. quantum query complexity. *Journal of Computer and System Sciences*, 72(2):220-238, 2006. Earlier versions at FOCS'2003 and quant-ph/0305028.

[2] A. Ambainis. Quantum walk algorithm for element distinctness. *SIAM Journal on Computing*, 37(1): 210-239, 2007. Also FOCS'04 and quant-ph/0311001.

[3] A. Ambainis, A. Bačkurs, J. Smotrovs and R. de Wolf. Optimal quantum query bounds for almost all Boolean functions. arXiv:1208.1122.

[4] R. Beals, H. Buhrman, R. Cleve, M. Mosca, and R. de Wolf. Quantum lower bounds by polynomials. *Journal of the ACM*, 48(4):778-797, 2001. Earlier versions at FOCS'1998 and quant-ph/9802049.

[5] C. H. Bennett, G. Brassard, C. Crépeau, R. Jozsa, A. Peres, W. K. Wootters, Teleporting an Unknown Quantum State via Dual Classical and Einstein-Podolsky-Rosen Channels, *Physical Review Letters*, 70:1895-1899, 1993.

[6] G. Brassard and P. Høyer. An exact quantum polynomial-time algorithm for Simon's problem. *Proceedings of the Israeli Symposium on Theory of Computing and Systems (ISTCS)*, pp. 12-23, 1997. Also quant-ph/9704027.

[7] G. Brassard, P. Høyer, M. Mosca, A. Tapp. Quantum amplitude amplification and estimation. In *Quantum Computation and Quantum Information Science*, AMS Contemporary Mathematics Series, 305:53-74, 2002. Also quant-ph/0005055.

[8] H. Buhrman, R. Cleve, A. Wigderson. Quantum vs. classical communication and computation. *Proceedings of STOC'98*, pp. 63Ű68. Also quant-ph/9802040.

[9] H. Buhrman, R. Cleve, S. Massar, R. de Wolf. Non-locality and Communication Complexity. *Reviews of Modern Physics*, 82:665-698, 2010. Also quant-ph/0907.3584

[10] H. Buhrman, R. de Wolf. Complexity measures and decision tree complexity: a survey. *Theoretical Computer Science*, 288:21-43, 2002.

[11] R. Cleve, A. Ekert, C. Macchiavello, M. Mosca. Quantum algorithms revisited. *Proceedings of the Royal Society of London A*, 454: 339-354, 1998.

[12] W. van Dam. Quantum oracle interrogation: Getting all information for almost half the price. In *Proceedings of FOCS'1998*, pp. 362–367. Also quant-ph/9805006.

[13] D. Deutsch. Quantum theory, the Church-Turing principle and the universal quantum computer. *Proceedings of the Royal Society in London A*, 400:97, 1985.

[14] D. Deutsch, R. Jozsa. Rapid solutions of problems by quantum computation. *Proceedings of the Royal Society of London A*, 439:553-558, 1992.

[15] E. Farhi, J. Goldstone, S. Gutman, A Quantum Algorithm for the Hamiltonian NAND Tree. *Theory of Computing*, 4:169-190, 2008. Also quant-ph/0702144.

[16] L. K. Grover. A fast quantum mechanical algorithm for database search. *Proceedings of STOC'96*, pp. 212-219.

[17] T. Hayes, S. Kutin, and D. van Melkebeek. The quantum black-box complexity of majority. *Algorithmica*, 34(4):480-501, 2002. quant-ph/0109101.

[18] P. Høyer, T. Lee, R. Špalek. Negative weights make adversaries stronger. *Proceedings of STOC'07*, pp. 526-535. Also quant-ph/0611054.

[19] B. Kalyanasundaram and G. Schnitger. The probabilistic communication complexity of set intersection. *SIAM Journal on Computing*, 5(4):545-557, 1992. Earlier version in Structures'87.

[20] E. Kushilevitz. Personal communication, cited in [25].

[21] G. Midrijānis. Exact quantum query complexity for total Boolean functions, quant-ph/0403168.

[22] A. Montanaro, R. Jozsa, G. Mitchinson. On exact quantum query complexity. arXiv:1111.0475.

[23] M. Nielsen, I. Chuang. *Quantum Computation and Quantum Information.* Cambridge University Press, 2000.

[24] N. Nisan and M. Szegedy. On the degree of Boolean functions as real polynomials. *Computational Complexity*, 4(4):301–313, 1994. Earlier version in STOC'92.

[25] N. Nisan, A. Wigderson. On rank vs. communication complexity. *Combinatorica*, 15: 557-565, 1995. Also FOCS'94.

[26] A. Razborov. On the distributional complexity of disjointness. *Theoretical Computer Science*, 106(2):385-390, 1992.

[27] B. Reichardt. Span Programs and Quantum Query Complexity: The General Adversary Bound Is Nearly Tight for Every Boolean Function. *Proceedings of FOCS'2009*, pp. 544-551. Also arXiv:0907.1622.

[28] B. Reichardt. Reflections for quantum query algorithms. *Proceedings of SODA'2011*, pp. 560-569. Also arXiv:1005.1601.

[29] B. Reichardt, R. Špalek. Span-program-based quantum algorithm for evaluating formulas. *Proceedings of STOC'2008*, pp. 103-112. Also arXiv:0710.2630.

[30] P. Shor. Algorithms for Quantum Computation: Discrete Logarithms and Factoring. *Proceedings of FOCS'94*, pp. 124-134. Also quant-ph/9508027.

[31] D. Simon. On the power of quantum computation. *SIAM Journal on Computing*, 26:1474-1483, 1997. Earlier version at FOCS'94.

[32] A. Vasilieva. Exact Quantum Query Algorithm for Error Detection Code Verification. *Proceedings of MEMICS'2009*, In *OpenAccess Series in Informatics (OASIcs)*, vol. 13, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. Also arXiv:0904.3660.

[33] R. de Wolf. Personal communication. November 6, 2012.